HW 1: Collaborative Filtering and Gradient Descent
Advanced Machine Learning

Prof: Yannet Interian

Due: Feb 2nd 2017

For this assignment you should submit the following:

- **hw-1.pdf** and **predictions.txt** to CANVAS.
- Your code to GITHUB cf.py (directory hw1).

hw-1.pdf should contain the answers to questions in 1.4, 1.6, 1.7, 18, 2. predictions.txt should have 12000 lines. It should be the output of

```
python cf.py  --train TrainingRatings.txt --test TestingRatingsMedium.txt
```

1 Implementing Collaborative Filtering for movie Ratings (20 points)

A typical data challenge for data science interviews is to give you some data and ask you to solve a modeling problem without using ML libraries. Often the problem can be modeled as a collaborative filtering. In this exercise you will practice how to solve this type of questions.

For this homework you are going to write a collaborative filtering algorithm to predict movie reviews ratings using a dataset of Netflix ratings.

1.1 Data

The dataset for this assignment is a subset of the movie ratings data from the Netflix Prize `http://www.netflixprize.com/`. It contains a training set, a test set, a movies file and a README file. You will use the ratings provided in the training set to predict those in the test set. You will compare your predictions with the actual ratings provided in the test set. The evaluation metrics you need to measure are the Mean Absolute Error and the Root Mean Squared Error. The README file is from the original set of Netflix files. It is included to comply with the terms of use for this data.

The files have the following format: MovieId,CustomerId,Rating

```
 head TestingRatingsMedium.txt
8,573364,1.0
8,2149668,3.0
8,1089184,3.0
8,2465894,3.0
8,534508,1.0
8,992921,4.0
8,595054,4.0
8,1298304,4.0
8,1661600,4.0
8,553787,2.0
```

1.2 User Similarity

For data on movie ratings we can use Pearson coefficient as a measure of similarity. Given users i and j, let I_i be the set of items that user i has rated.

$$w_{ij} = \frac{\sum_{k \in I_i \cap I_j} (R_{ik} - \bar{R}_i)(R_{jk} - \bar{R}_j)}{\sqrt{\sum_{k \in I_i \cap I_j} (R_{ik} - \bar{R}_i)^2 \sum_{k \in I_i \cap I_j} (R_{jk} - \bar{R}_j)^2}}$$

Where $\bar{R}_i = \frac{1}{|I_i|} \sum_{k \in I_i} R_{ik}$ is the average rating for user i.

If $I_i \cap I_j = \emptyset$ let's define $w_{ij} = 0$.

1.3 Predict Movie Ratings

Let R_{jk} be the rating of user j on item/movie k. For user i and movie k we can predict ratings by using the following formula:

$$\hat{R}_{ik} = \bar{R}_i + \frac{1}{\sum_{j \in U_k} |w_{ij}|} \sum_{j \in U_k} w_{ij}(R_{jk} - \bar{R}_j)$$

Where n denotes the number of users, $\bar{R}_i$ is the average rating for user i, w_{ij} are similarity metrics between user i and j and U_k are the set of users that rated movie k

1.4 Evaluation

Include RMSE and MAE in the pdf
Compute mean absolute error and the root mean squared error on the test set as the evaluation metrics. I get the following results on this dataset

```
python cf.py  --train TrainingRatings.txt --test TestingRatingsMedium.txt
RMSE 0.8899
MAE 0.6975
```

1.5 Implementing Collaborative Filtering algorithm

Implement the collaborative filtering algorithm following the rules below.

- Use the template I am giving you to write your code. I am also giving you a test script that you should use. For the test script to work you need to write the functions given on the template (*cf.py.*)
- Given path to a training and testing files, I should be able to run your code using the command line:

  ```
  python cf.py --train <path>/training.txt  --test <path>/testing.txt
  ```

- To run the test script

```
python cf_test.py
```

- Your code should output a text file called "predictions.txt" with 4 columns. The same columns as the test file and your prediction in the last column.
- Your code should print the *mean absolute error* and the *root mean squared error* to the standard output.
- Your prediction should have at least two significant digits e.g. 4.5.
- Do not use any machine learning libraries.
- Implementation tricks:
  - Do not compute similarity matrix for all users. It would exceed your memory limit. Compute similarities as needed to make predictions.
  - Implement your algorithm using the small dataset provided. After you are done run it on the large dataset.

1.6 The Cold Start Problem. (2 points)

Discuss this in the pdf
What prediction would you use in these cases:

- A new user but a known movie
- A new movie and a known user
- A new user and new movie

1.7 Complexity (1 point)

Discuss this in the pdf
What is the complexity of this algorithm (explain)? Here you can make simplifying assumptions. For example, start by analyzing the complexity of computing similarity between two users assuming that users rated every movie.

1.8 Collaborative Filtering for Pandora. (2 points)

Discuss this in the pdf
Write pseudocode describing an algorithm (similar to the one in previous section) that would do collaborative filtering in a dataset from Pandora, Spotify or YouTube. Assume you just have users and songs/videos these users have played but not explicit ratings. Why the algorithm from the previous section doesn't work?

1.9 Rules for Grading

If your code doesn't run in my computer I wouldn't give partial credits (the code it worth 15 points). Make sure you don't hardcode paths and you follow the instructions.

2 Gradient Descent for Matrix Factorization (10 points)

Discuss this in the pdf
We want to extend the Matrix Factorization model discussed in class to add a "bias" parameter for each user and another "bias" parameter for each movie. For the problem in class we had the parameters matrix U and V, we are adding u^0 which is a vector of dimension n_u and v^0 which is a vector of dimension n_m. The equations

$$\hat{y}_{ij} = u_i^0 + v_j^0 + u_i \cdot v_j$$

Write the gradient descent equations for this problem. You can write the equation in latex or take a picture that can be included in the pdf. Make sure your formulas are readable. I am providing a latex template.